#SHAREorg

Using WebSphere Application Server Optimized Local Adapters (WOLA) to Integrate COBOL and zAAP-able Java

David Follis
IBM

March 12, 2014
Session Number 14693

SHARE in Anaheim

Trademarks

Disclaimer

- The information contained in this documentation is provided for informational purposes only. While efforts were many to verify the completeness and accuracy of the information contained in this document, it is provided “as is” without warranty of any kind, express or implied.
- This information is based on IBM’s current product plans and strategy, which are subject to change without notice. IBM will not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation.
- Nothing contained in this documentation is intended to, nor shall have the effect of , creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of the IBM software.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

WebSphere Application Server on System Z

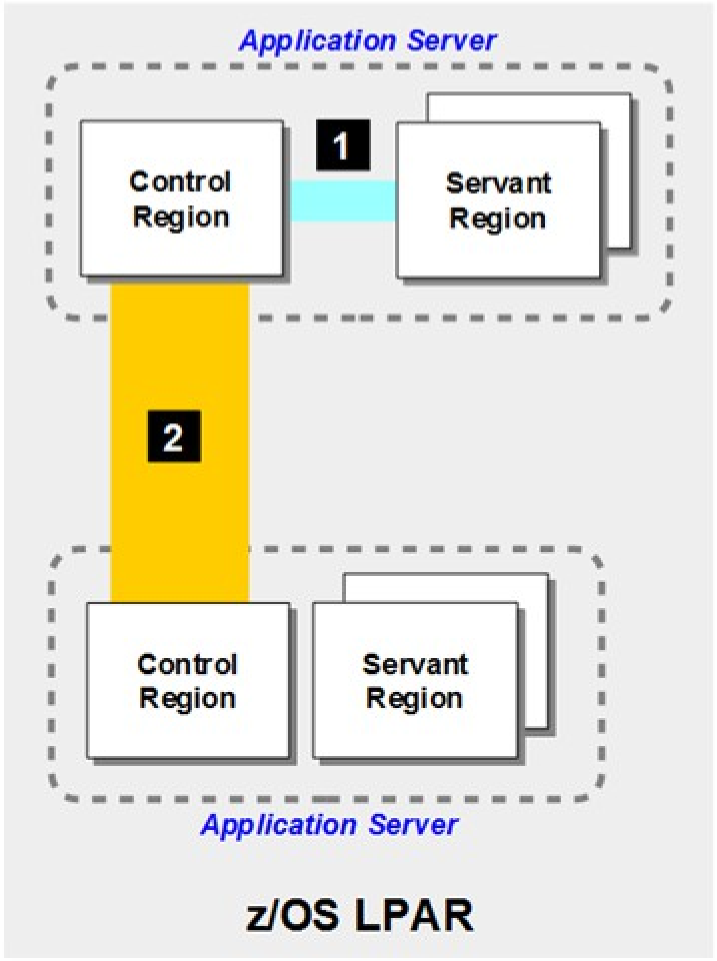
Session	Title	Time	Room	Speaker
14618	Getting Started with WebSphere Liberty Profile on z/OS	Monday 9:30	Grand Ballroom Salon C	Loos/Follis
14692	Getting Started with WebSphere Compute Grid	Tuesday 9:30	Grand Ballroom Salon J	Hutchinson/Loos
14693	Using WebSphere Application Server Optimized Local Adapters (WOLA) to Migrate Your COBOL to zAAP-able Java	Wednesday 9:30	Grand Ballroom Salon K	David Follis
14620	WebSphere Liberty Profile on Windows AND z/OS (among other things) Hands-on Lab	Wednesday 1:30	Platinum Ballroom Salon 7	
14949	Tips Learned Implementing Websphere Application Server (WAS) on Linux for IBM System z	Wednesday 3:00	Grand Ballroom Salon G	Eberhard Pasch
14709	Need a Support Assistant? Check Out IBM's! (ISA)	Thursday 8:00	Grand Ballroom Salon A	Mike Stephen
15050	z/OSMF 2.1 Implementation and Configuration	Thursday 8:00	Grand Ballroom Salon G	Greg Daynes
14832	Web Apps using Liberty Profile Technology in CICS	Thursday 11:00	Platinum Ballroom Salon 2	Ian Mitchell
14722	Assimilating WebSphere Application Server into your z/OS WLM Configuration	Thursday 1:30	Orange County Salon 1	David Follis
15017	Using IBM WebSphere Application Server and IBM WebSphere MQ Together [z/OS & Distributed]	Thursday 3:00	Grand Ballroom Salon A	Ralph Bateman

WebSphere Optimized Local Adapters

Introduction and Overview

In The Beginning: Cross Memory

WAS z/OS has had cross-memory from the beginning:



1 – Controller to Servant

A request queued to WLM is pulled from the CR to the SR using cross-memory services. What gets queued in WLM is really a token to the address location within the CR where the request is currently held.

2 – Server to Server

When a servlet in one server makes an IIOP call to an EJB in another server in the same cell and on the same LPAR, WAS z/OS recognizes this and bypasses the TCP stack. The IIOP call is made using cross-memory services.

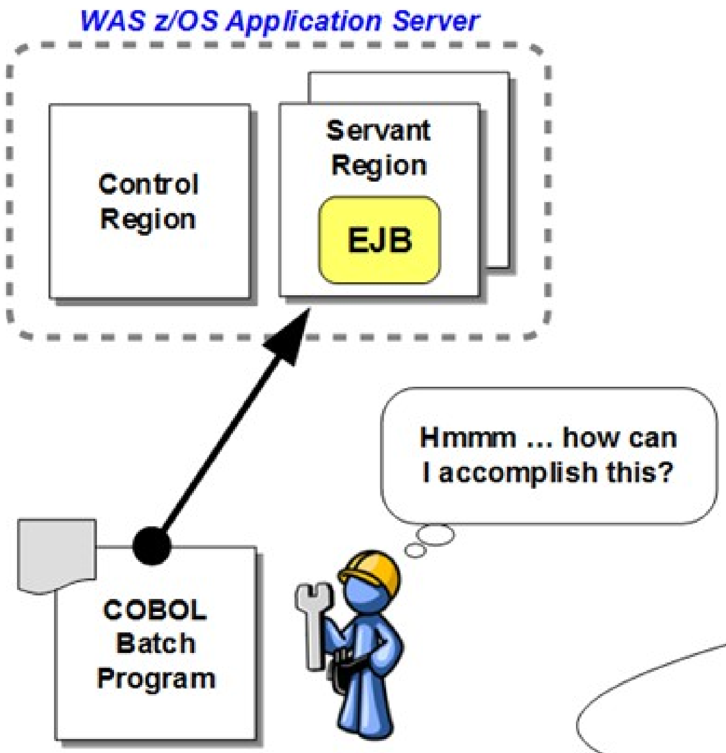
This is called LOCALCOMM, and it's the key to starting the story of what WOLA is and how it works.

We're setting the stage for WOLA by providing historical context

WOLA is built on this from-the-beginning LOCALCOMM function

Original Requirement: Batch into WAS

Access Java asset in WAS from batch COBOL


WAS z/OS Application Server
Control Region
Servant Region
EJB
COBOL Batch Program
Hmmm ... how can I accomplish this?

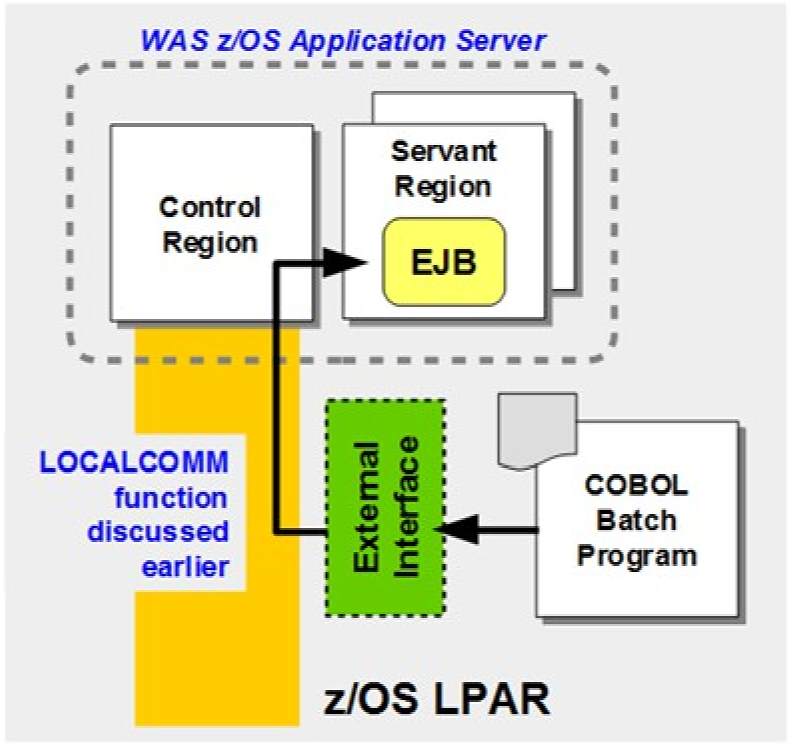
Potential Solutions:

- Use JNI and RMI/IIOP
  Somewhat complex: require C/C++ stub for JNI; JNI code to interface to Java; JVM to host Java used to RMI/IIOP call into WAS z/OS; code to perform bootstrap and lookup, etc., etc.
- Use MQ
  Most obvious ... but would require target EJB to be MDB or know about MQ to PUT/GET; introduces an asynchronous nature that may not suit batch program.
- Use Web Services
  Challenging unless batch program has infrastructure to open sockets, understand HTTP, form SOAP or RESTful calls. Plus, web services is somewhat high-overhead with larger latency per call, which can harm batch processing window.
- Something else?

This is where WOLA came from

Externalized LOCALCOMM = WOLA

WOLA is programmatic interface to LOCALCOMM



The cross-memory function already existing within WAS z/OS

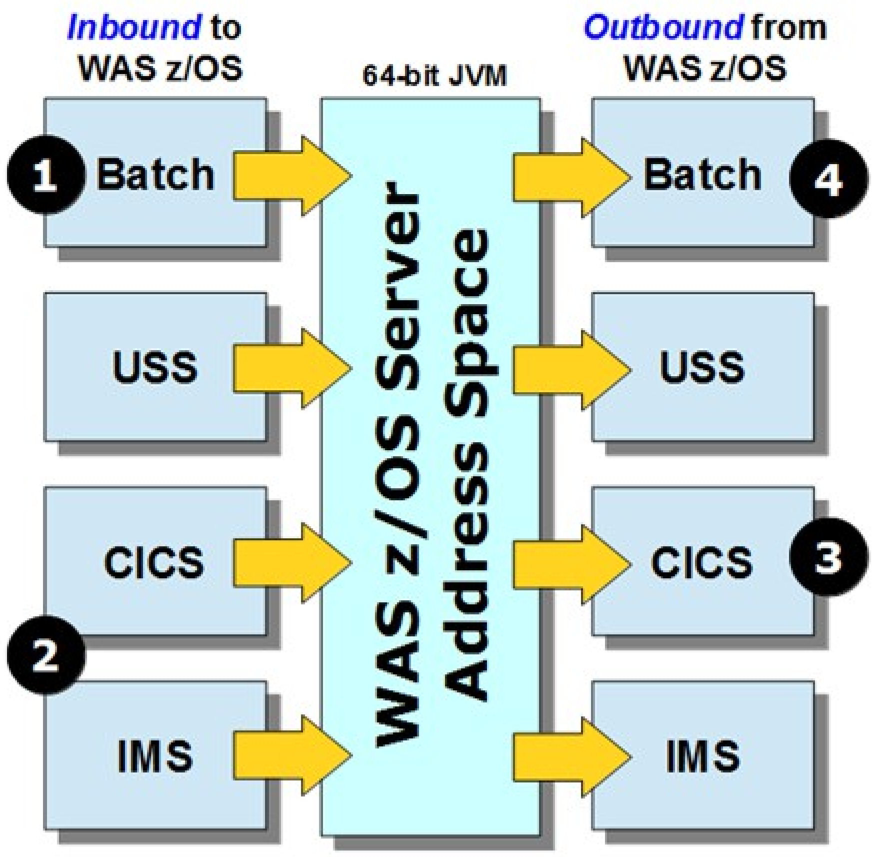
Key was to provide a way for external address spaces to access it

WOLA is just that – an infrastructure and interface for external address spaces to participate in the LOCALCOMM exchange mechanism of WAS z/OS

★ Not *just* COBOL batch … that's just our starting point. Supports several different programming languages, CICS and IMS

WOLA is Bi-Directional

This chart shows a few use-cases



1 – Inbound from Batch

This is the use-case we said sparked the development of WOLA. We see this in cases where batch programs need data from EJB assets, particularly ISV apps

2 – Inbound from CICS or IMS

This is in many ways the same as inbound from batch, though with CICS or IMS you have the added benefit of transaction and security propagation.

3 – Outbound to CICS

The key benefit of this is it allows channels/containers over a cross-memory exchange. CTG EXCI is limited to 32K COMMAREA, and CTG channels/containers provided over IPIC.

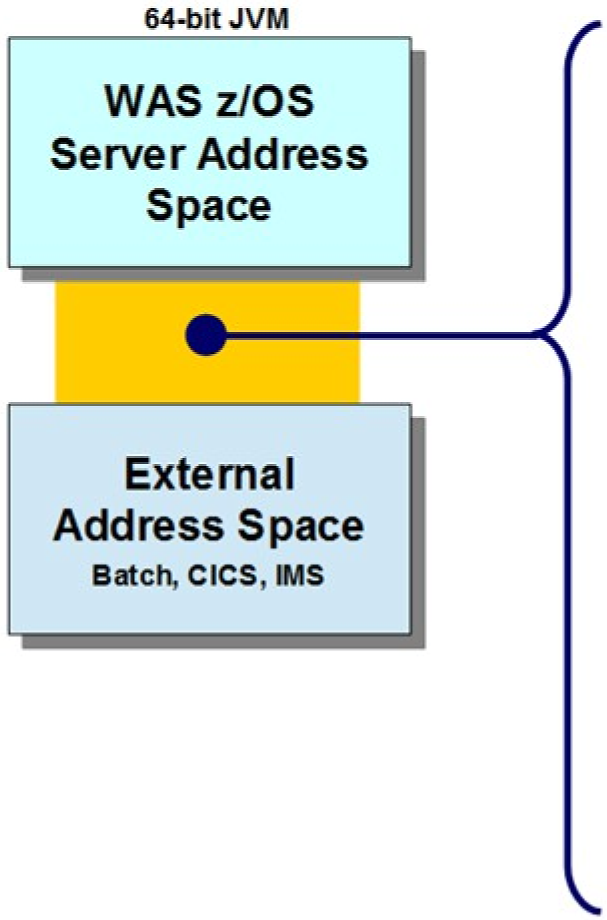
Note: CTG has other benefits related to HA designs and cross-LPAR communications. We are *not* saying WOLA replaces CTG; it complements it.

4 – Outbound to Batch

This is an interesting use-case we're seeing more of lately ... it provides a means of exposing existing batch programs as a service, using web services or any other access method supported by WAS itself.

Value of WOLA

WOLA has several strong technical value attributes



Low Latency

Bypasses TCP stack; LOCALCOMM code path relatively lightweight; lowest per-call latency available

Large Messages

Provides efficient means of passing large messages (up to 2GB) including CICS channels/containers and IMS multi-segment/large-message support; avoids overhead of segmenting and rebuilding messages.

Secure Exchange

Cross-memory never touches network stack or network infrastructure so it can't be sniffed and it can't be altered in flight.

Transaction Propagation

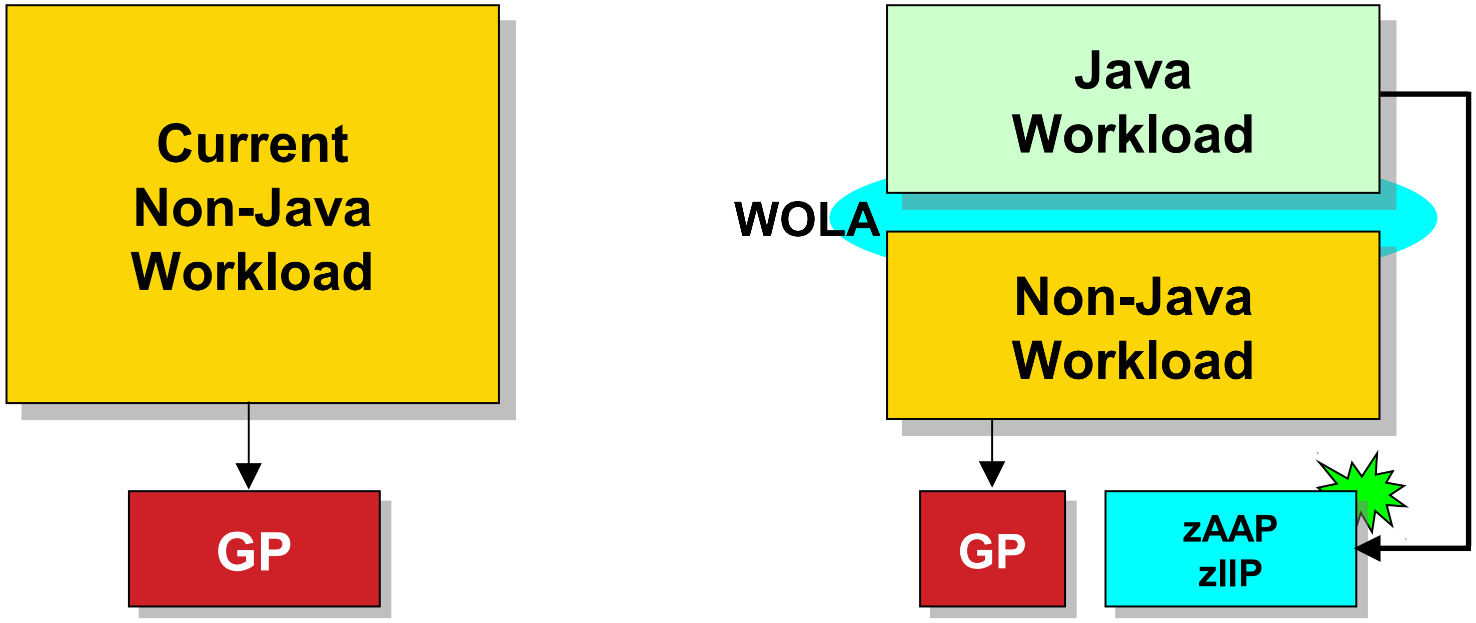
Ability to propagate TX with CICS or IMS so global transactions using z/OS RRS as synchpoint coordinator

Security Propagation

Ability to propagate identity with CICS or IMS without requiring coding of aliases

Integrating Java / Non-Java Using WOLA

Enabling some of your GP work to be refactored to Java



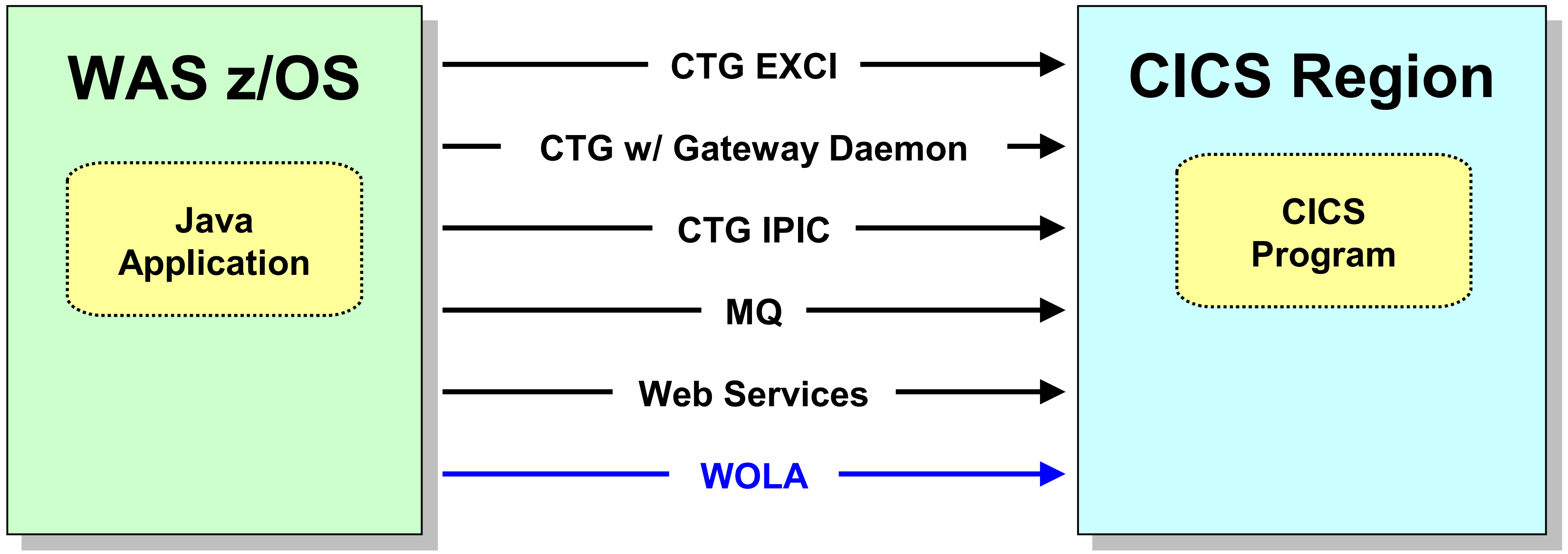
WOLA is a technology that enables integration of Java and non-Java assets in a way that provides high throughput and security

WebSphere Optimized Local Adapters

Outbound WOLA to CICS

Accessing CICS from WAS z/OS

Many different ways this can be done



Some considerations:

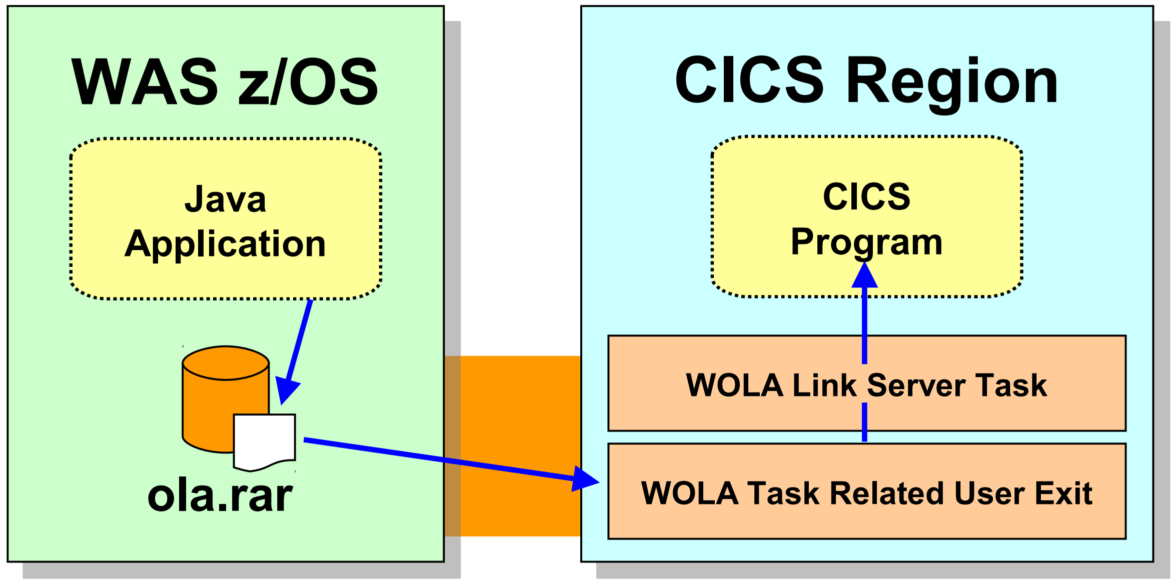
- CTG is a separately licensed product
- EXCI limited to 32K COMMAREA
- CTG IPIC allows >32K and involves TCP/IP
- MQ is asynchronous
- Web Services may have higher overhead

WOLA:

- Comes with WAS z/OS
- COMMAREA or Chan/Containers
- TX and Security Propagation
- Cross Memory latency

WOLA Support and CICS

Here's what outbound WOLA to CICS looks like



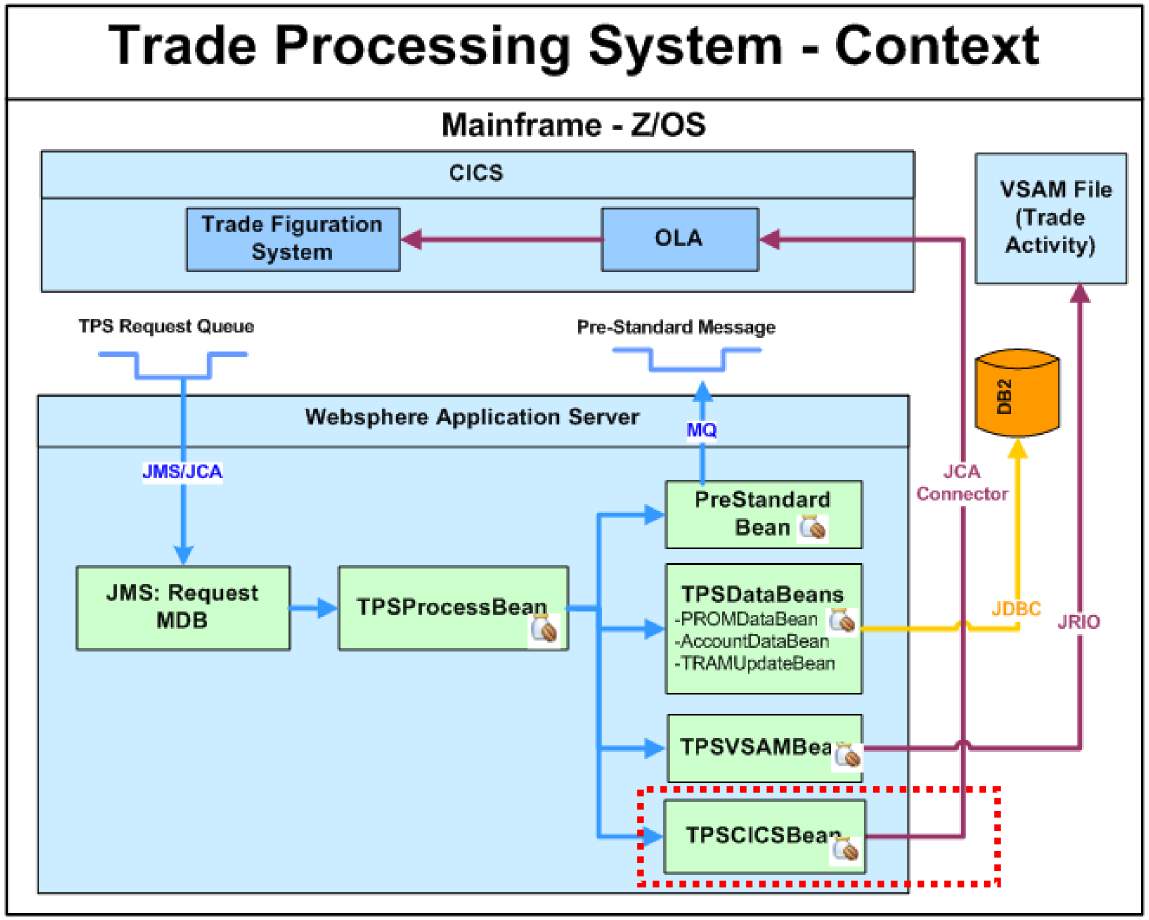
WOLA TRUE and Link Server provide support in CICS region and hides target CICS from any knowledge of WOLA

WOLA supplies a JCA resource adapter, which application uses to access WOLA services

WOLA "registration" is the cross-memory connection over which communications flow

Customer Use-Case Example

WOLA as part of complex stock trading architecture



WOLA used as one part of larger-scale application topology

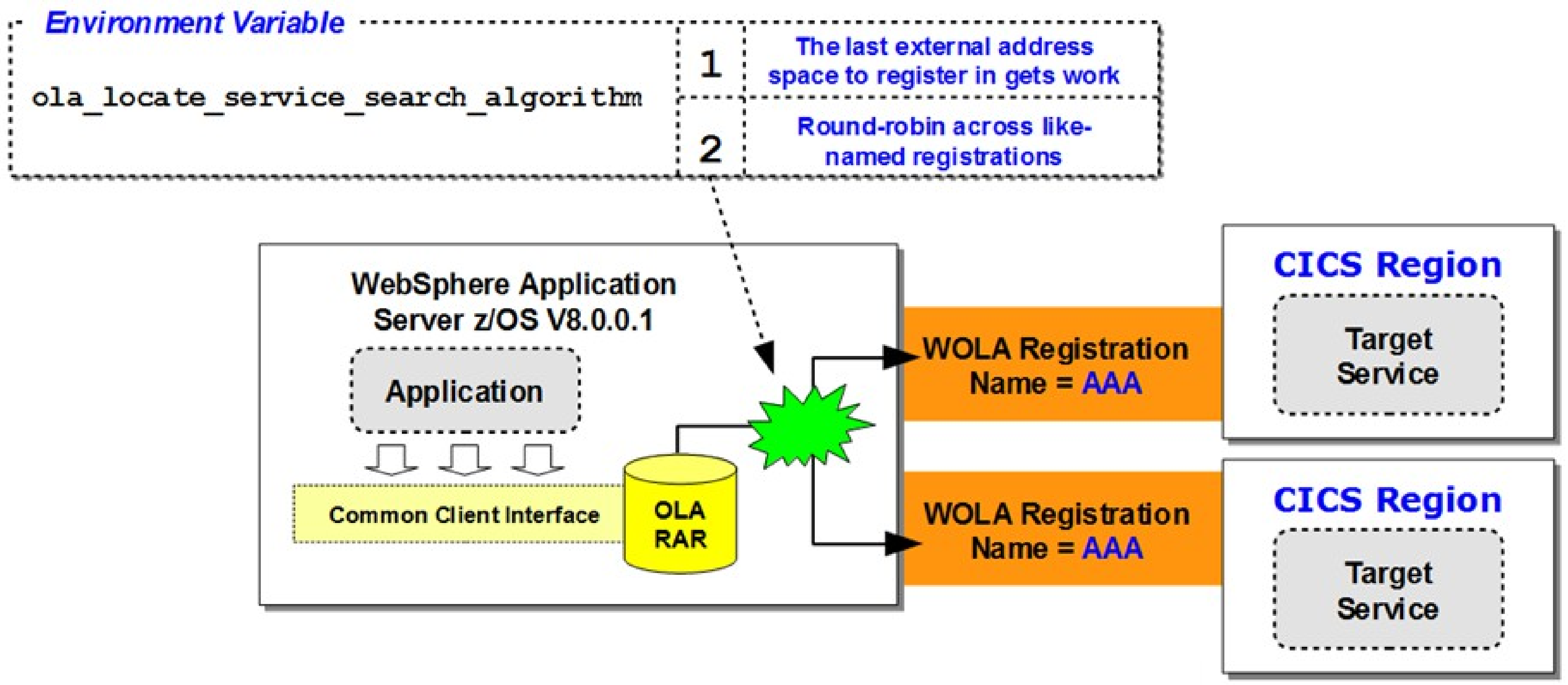
Proof of Concept evaluated WOLA compared to CTG

WOLA shows better performance for COMMAREA and Channels/Containers

Eliminates need for CTG for this part of the architecture

Multiple CICS Regions (Gateways)

WOLA supports multiple concurrent registrations


Environment Variable
ola_locate_service_search_algorithm
1
The last external address space to register in gets work
2
Round-robin across like-named registrations
WebSphere Application Server z/OS V8.0.0.1
Application
Common Client Interface
OLA RAR
WOLA Registration Name = AAA
WOLA Registration Name = AAA
CICS Region
Target Service
CICS Region
Target Service

WOLA to multiple CICS regions serving as gateway regions to larger CICSPlex

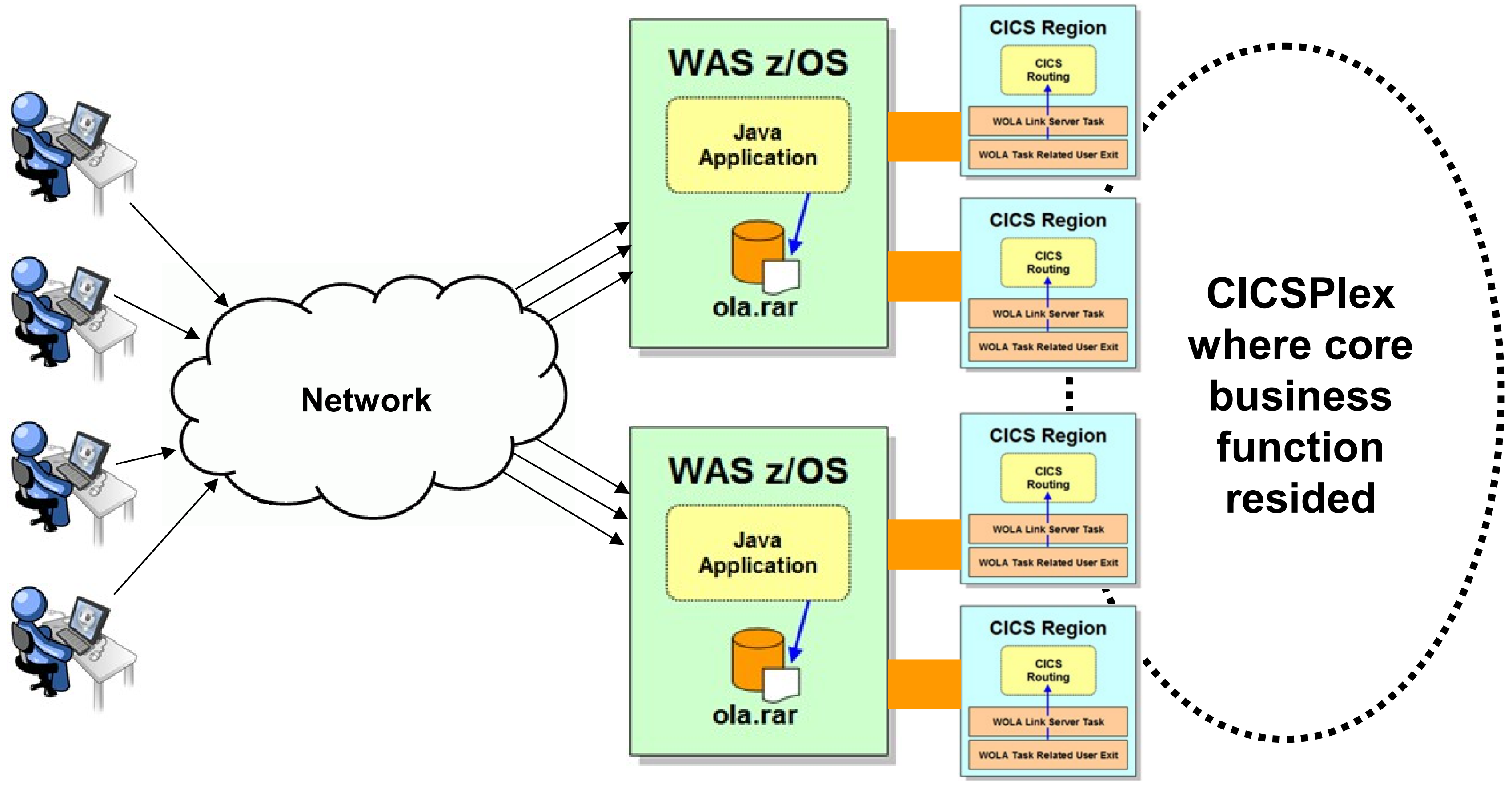
Round-robin support added in 8.0.0.1

Provides highly utilized WOLA connections with balancing and HA

SHARE in Anaheim

Customer Use-Case Example

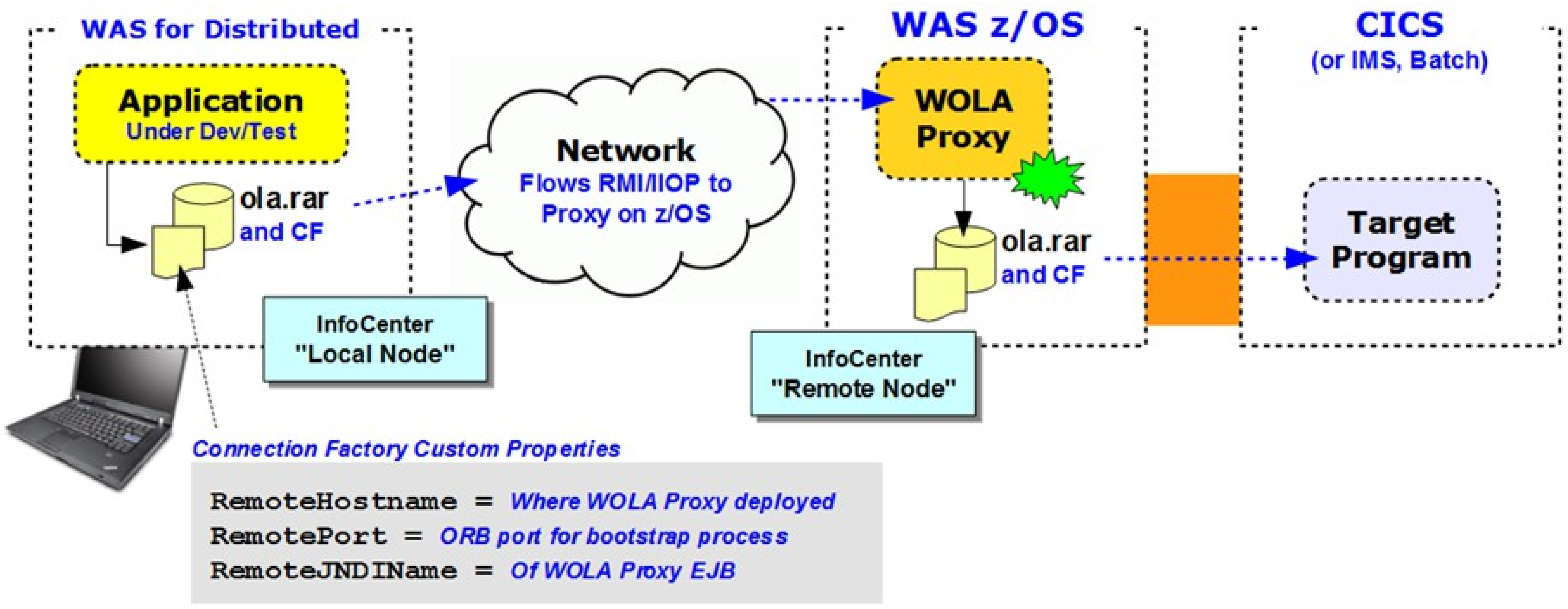
Round-robin WOLA to CICS Gateway Regions



Network clients access business functions through WAS z/OS, which turns and drives CICS function over WOLA implemented with round-robin support to CICS gateway regions

Development Mode Support

Allows Java developers greater dev/test flexibility



```
RemoteHostname = Where WOLA Proxy deployed
RemotePort = ORB port for bootstrap process
RemoteJNDIName = Of WOLA Proxy EJB
```

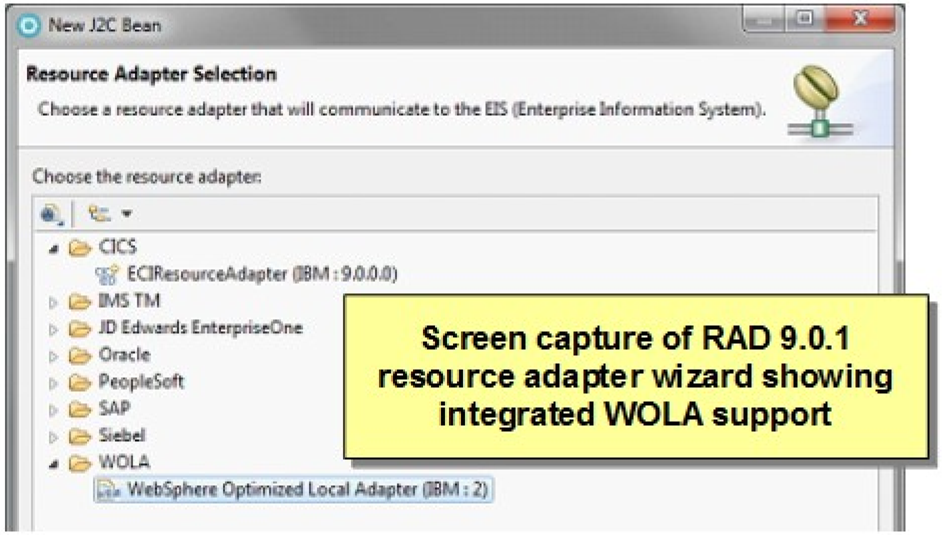
Developer is free to deploy, test, change settings on development workstation WAS

To application WOLA is in use, but in reality call is mapped to network

WOLA Proxy EJB intercepts and makes “real” WOLA call to CICS

WOLA and IBM RAD 9.0.1

RAD 9.0.1 includes integrated tooling support for WOLA


New J2C Bean

Resource Adapter Selection

Choose a resource adapter that will communicate to the EIS (Enterprise Information System).

Choose the resource adapter:

- CICS
  - ECIResourceAdapter (IBM : 9.0.0.0)
- IMS TM
- JD Edwards EnterpriseOne
- Oracle
- PeopleSoft
- SAP
- Siebel
- WOLA
  - WebSphere Optimized Local Adapter (IBM : 2)

Screen capture of RAD 9.0.1 resource adapter wizard showing integrated WOLA support

Note: This new RAD 9.0.1 WOLA support applies to WAS 8.5.0.+ levels, which is the level embedded with V9 RAD. The RAD support can be used on prior levels of WAS z/OS, but users will need to bring in the associated level's ola_apis.jar and ola.rar resource adapter and place it on their RAD build path.

IBM Webpage with details of RAD 9.0.1 Features and Functions

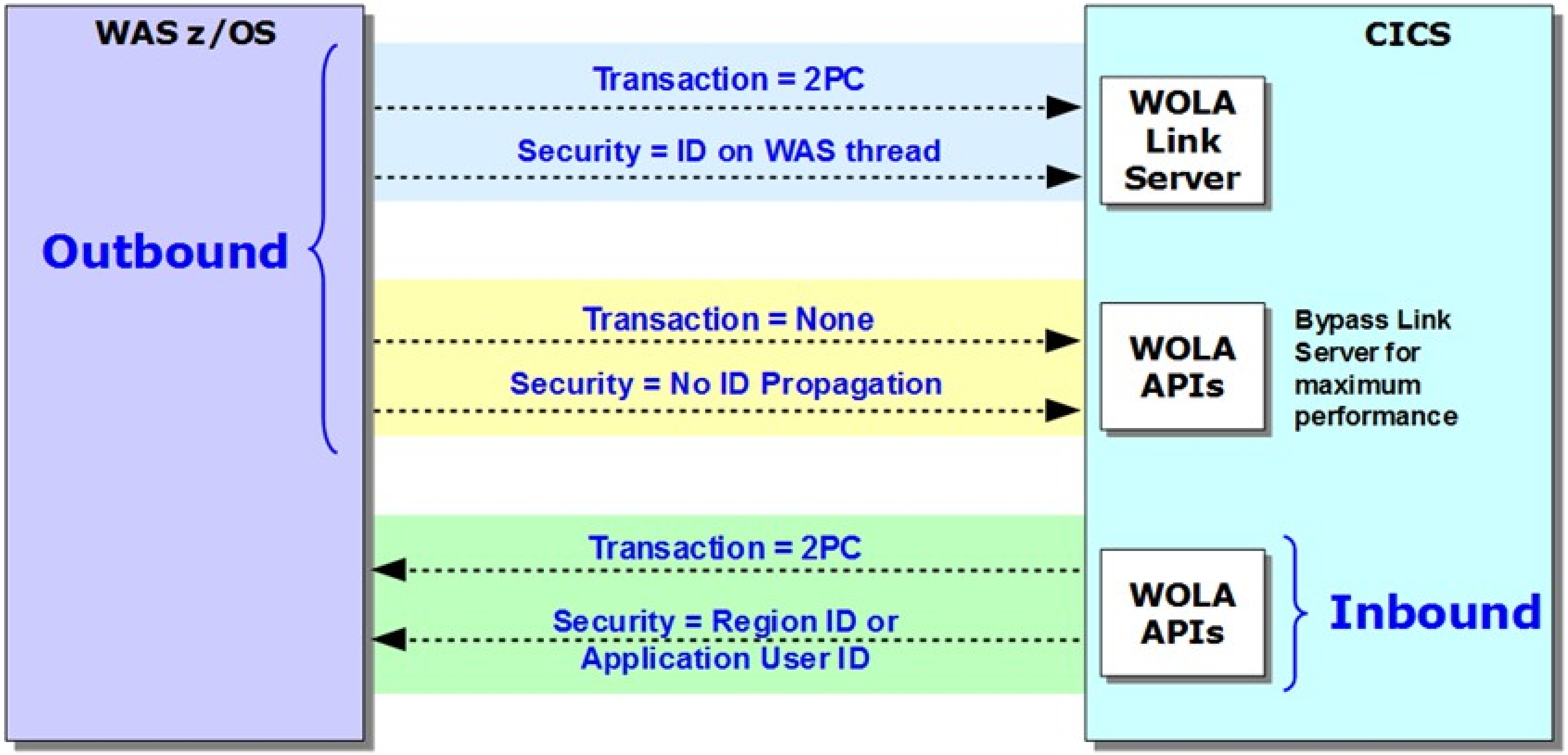
`http://www.ibm.com/support/docview.wss?uid=swg27038836`

IBM developerWorks article showing end-to-end use case scenario

`http://www.ibm.com/developerworks/websphere/techjournal/1312_mulvey/1312_mulvey.html`

WOLA and CICS Summary Chart

One chart showing supported functionality



WAS z/OS
Outbound
Transaction = 2PC
Security = ID on WAS thread
Transaction = None
Security = No ID Propagation
Transaction = 2PC
Security = Region ID or Application User ID
CICS
WOLA Link Server
WOLA APIs
Bypass Link Server for maximum performance
WOLA APIs
Inbound

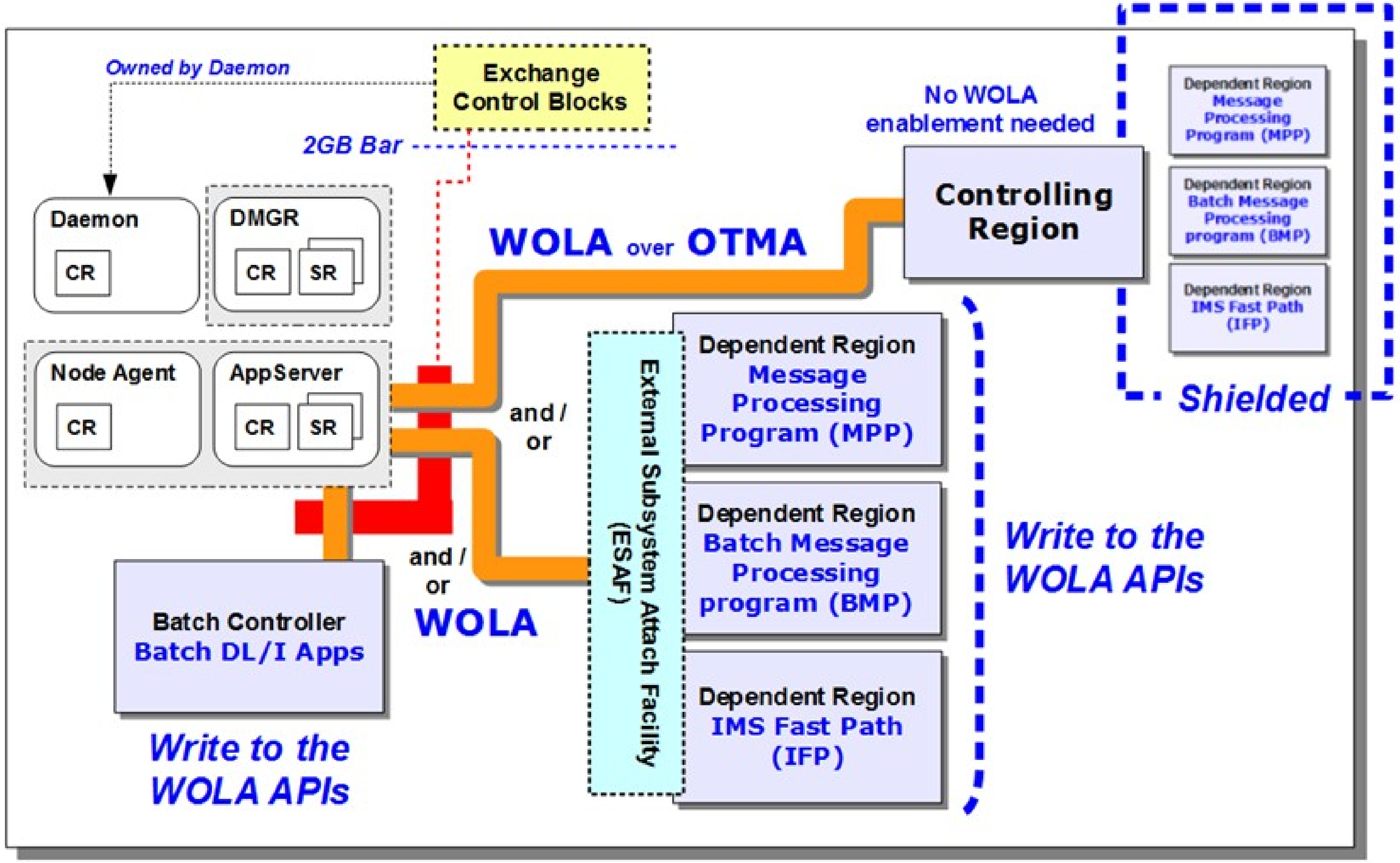
The registration into WAS must have the appropriate TXN and SEC settings to support propagation of global transaction and propagation of security identity

WebSphere Optimized Local Adapters

WOLA and IMS

WOLA Works with IMS

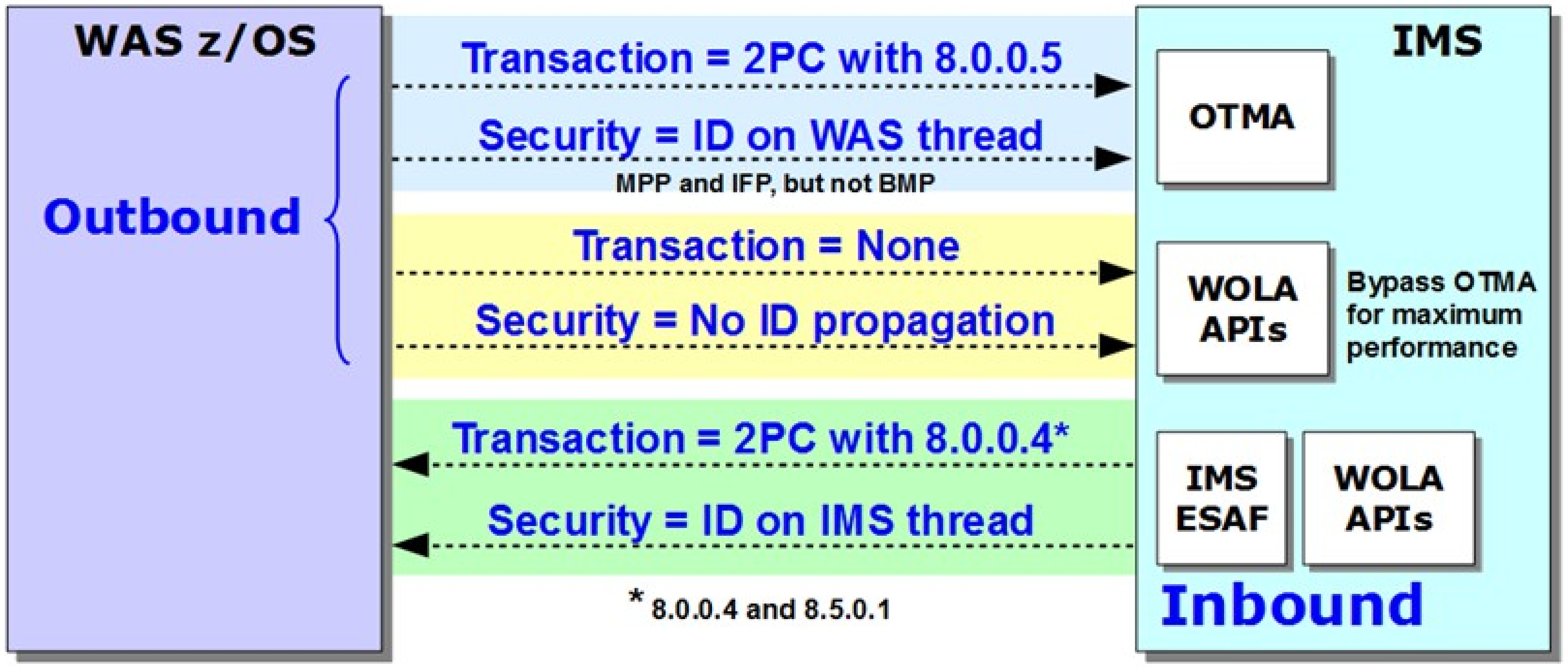
Here's a quick summary chart



Owned by Daemon
Exchange Control Blocks
2GB Bar
Daemon
CR
DMGR
CR
SR
Node Agent
CR
AppServer
CR
SR
WOLA over OTMA
No WOLA enablement needed
Controlling Region
Dependent Region Message Processing Program (MPP)
Dependent Region Batch Message Processing program (BMP)
Dependent Region IMS Fast Path (IFP)
Shielded
and / or
and / or
WOLA
External Subsystem Attach Facility (ESAF)
Dependent Region Message Processing Program (MPP)
Dependent Region Batch Message Processing program (BMP)
Dependent Region IMS Fast Path (IFP)
Write to the WOLA APIs
Batch Controller
Batch DL/I Apps
Write to the WOLA APIs

Summary of WOLA Capabilities and IMS

Particularly TX and Security Assertion



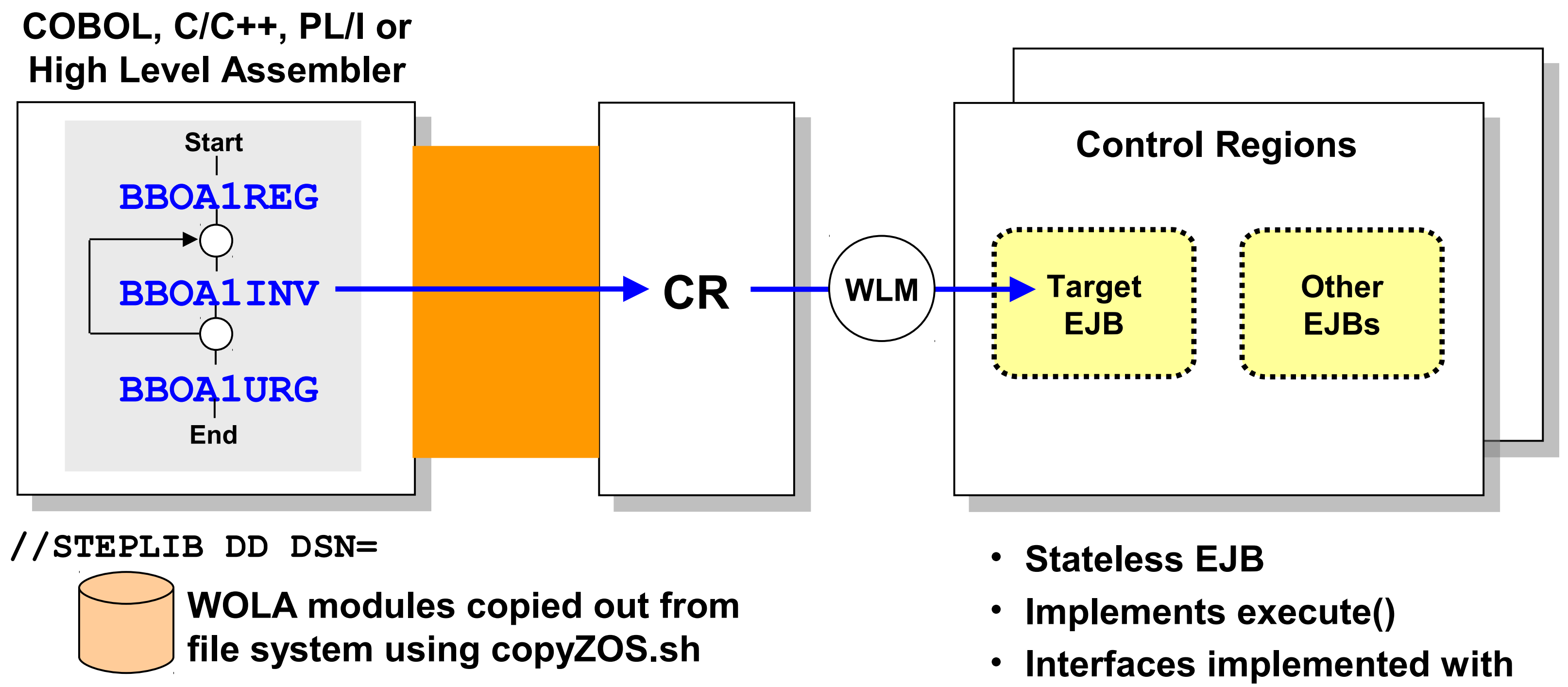
WAS z/OS
Outbound
Transaction = 2PC with 8.0.0.5
Security = ID on WAS thread
MPP and IFP, but not BMP
Transaction = None
Security = No ID propagation
Transaction = 2PC with 8.0.0.4*
Security = ID on IMS thread
* 8.0.0.4 and 8.5.0.1
IMS
OTMA
WOLA APIs
Bypass OTMA for maximum performance
IMS ESAF
WOLA APIs
Inbound

WebSphere Optimized Local Adapters

Inbound WOLA and APIs

Key Elements of the Inbound Design

Target EJB and use of the WOLA APIs



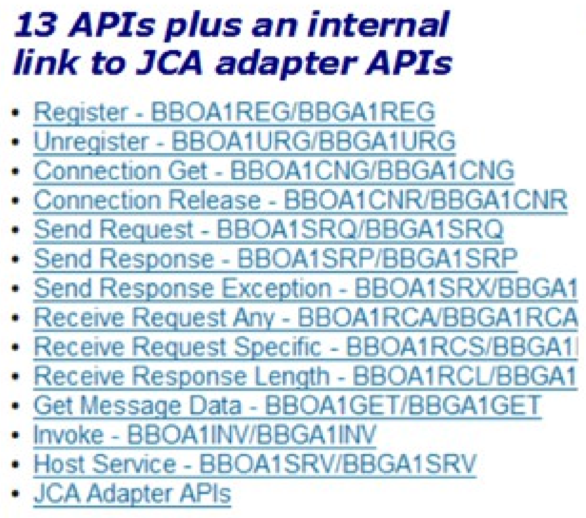
- **Stateless EJB**
- **Implements execute()**
- **Interfaces implemented with supplied WOLA classes**

The program outside of WAS uses the WOLA APIs to register and invoke the target EJB and get a response. The target EJB has a few easy-to-implement requirements. That EJB may call other EJBs as part of its processing.

WOLA API InfoCenter Article

An excellent reference for the WOLA APIs

InfoCenter `cdat_olaapis`

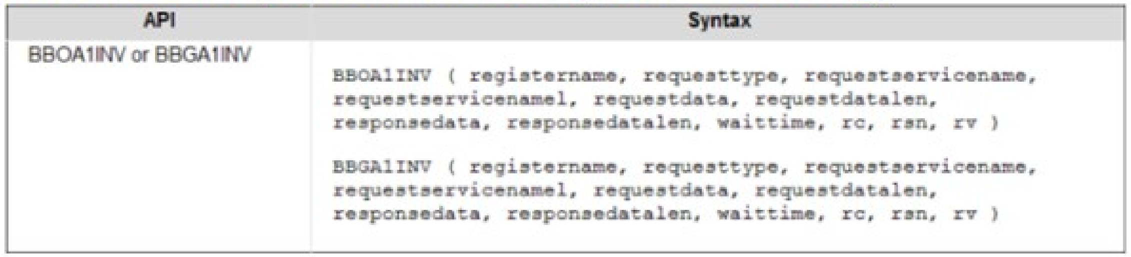


13 APIs plus an internal link to JCA adapter APIs

- Register - BBOA1REG/BBGA1REG
- Unregister - BBOA1URG/BBGA1URG
- Connection Get - BBOA1CNG/BBGA1CNG
- Connection Release - BBOA1CNR/BBGA1CNR
- Send Request - BBOA1SRQ/BBGA1SRQ
- Send Response - BBOA1SRP/BBGA1SRP
- Send Response Exception - BBOA1SRX/BBGA1
- Receive Request Any - BBOA1RCA/BBGA1RCA
- Receive Request Specific - BBOA1RCS/BBGA1
- Receive Response Length - BBOA1RCL/BBGA1
- Get Message Data - BBOA1GET/BBGA1GET
- Invoke - BBOA1INV/BBGA1INV
- Host Service - BBOA1SRV/BBGA1SRV
- JCA Adapter APIs

APIs that start with BBO* are 31-bit callable; BBG* are 64-bit callable

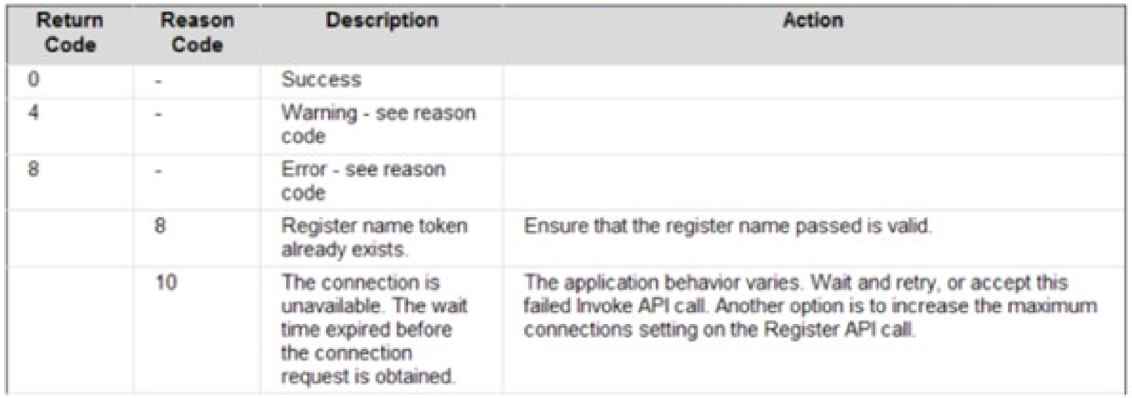
Parameter map (with full descriptions following)



API	Syntax

BBOA1INV or BBGA1INV	`BBOA1INV ( registername, requesttype, requestservicename, requestservicenamel, requestdata, requestdatalen, responsedata, responsedatalen, waittime, rc, rsn, rv )`
	
	`BBGA1INV ( registername, requesttype, requestservicename, requestservicenamel, requestdata, requestdatalen, responsedata, responsedatalen, waittime, rc, rsn, rv )`

Return Code / Reason Code descriptions for each API

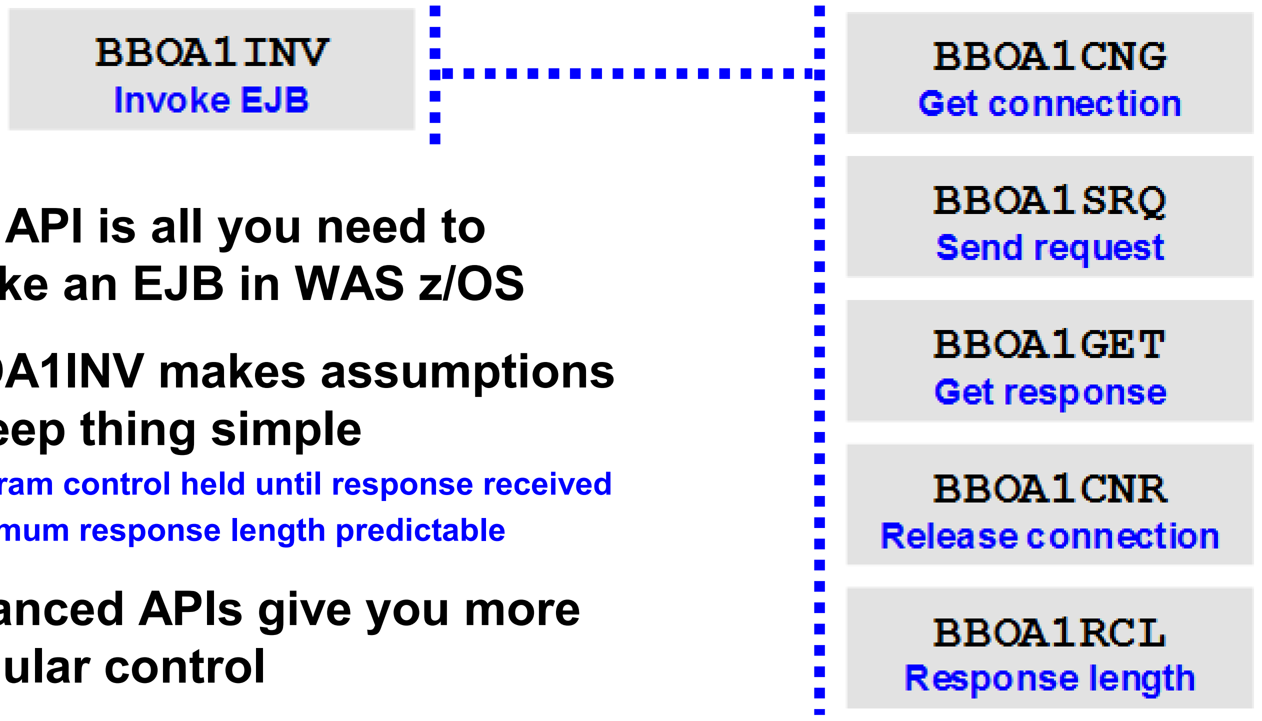


Return Code	Reason Code	Description	Action
0	-	Success	
4	-	Warning - see reason code	
8	-	Error - see reason code	
	8	Register name token already exists.	Ensure that the register name passed is valid.
	10	The connection is unavailable. The wait time expired before the connection request is obtained.	The application behavior varies. Wait and retry, or accept this failed Invoke API call. Another option is to increase the maximum connections setting on the Register API call.

This article spells out the details of each of the 13 WOLA APIs. However, it implies all 13 need to be used, which is not true. Over the next two charts we'll see how to organize the APIs into categories of usage

APIs … Basic vs. Advanced

Basic APIs are simple to use but make assumptions



One API is all you need to invoke an EJB in WAS z/OS

BBOA1INV makes assumptions to keep thing simple

- Program control held until response received
- Maximum response length predictable

Advanced APIs give you more granular control

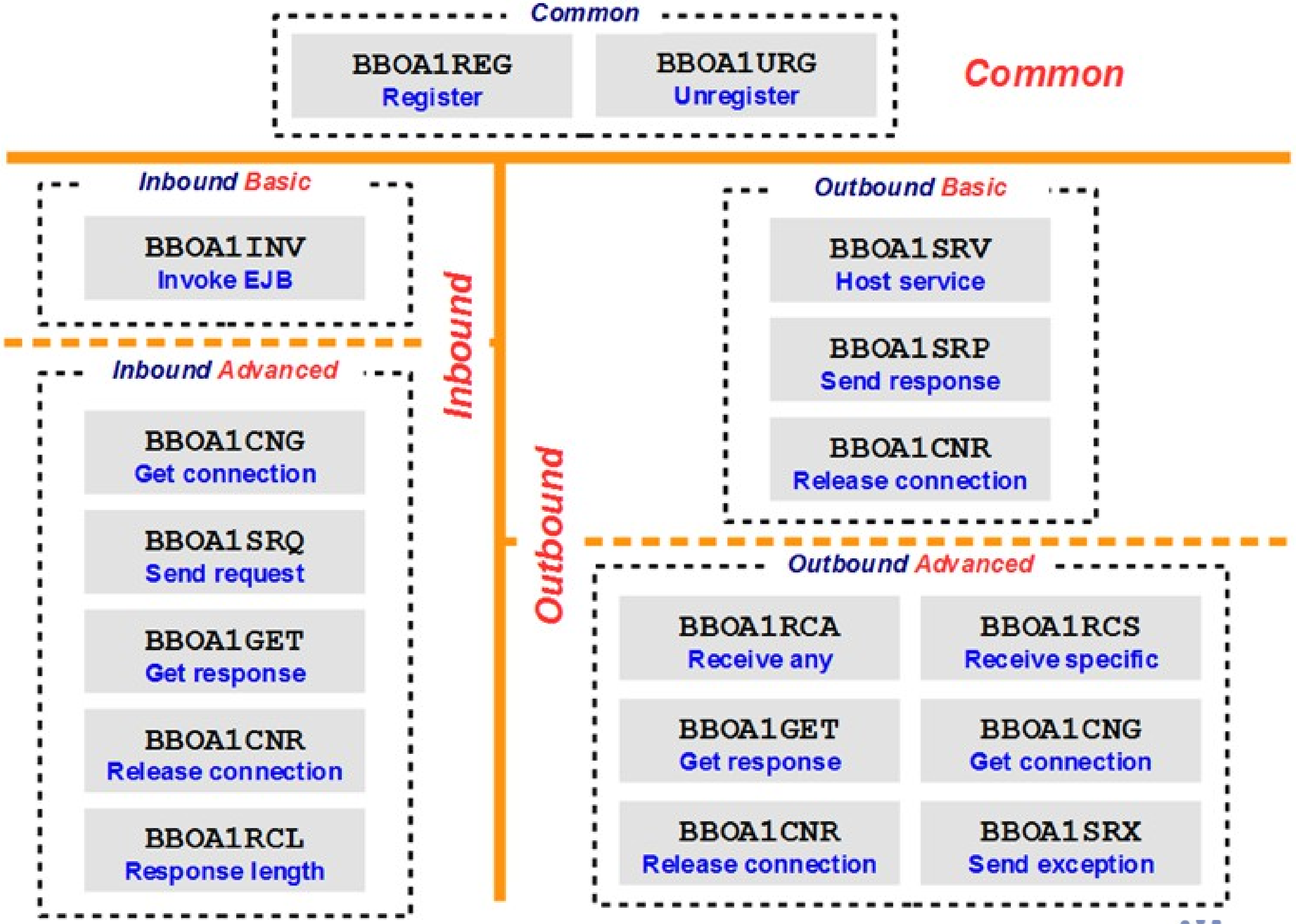
Advanced APIs do the same function as basic, but with more control given to you

If the "basic" APIs suit your needs, then use them. But if you need more control, then use "advanced." At the end of the day the same result is achieved – use of WOLA to communicate between address spaces

How the APIs Organize

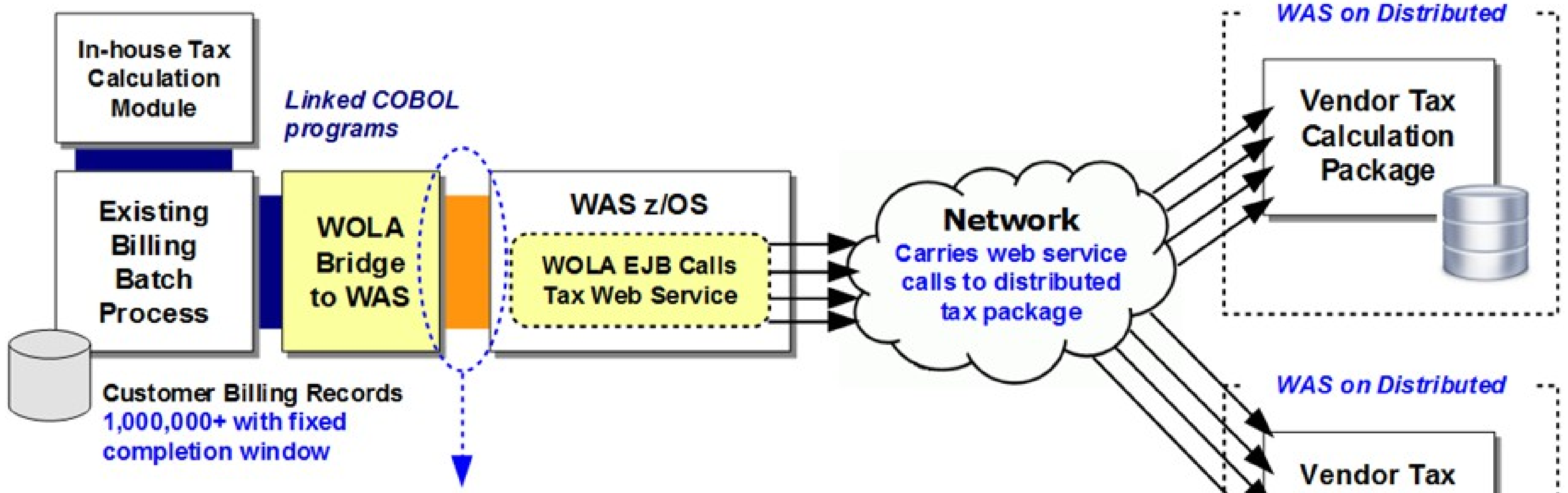
Inbound, outbound … basic and advanced

- BBOA1REG
- BBOA1URG
- BBOA1CNG
- BBOA1CNR
- BBOA1SRQ
- BBOA1SRP
- BBOA1SRX
- BBOA1RCA
- BBOA1RCS
- BBOA1RCL
- BBOA1GET
- BBOA1INV
- BBOA1SRV



Tax Calculation on Web Service Platform

Real example of WOLA usage with advanced APIs



In-house Tax Calculation Module
Linked COBOL programs
Existing Billing Batch Process
WOLA Bridge to WAS
WAS z/OS
WOLA EJB Calls Tax Web Service
Network
Carries web service calls to distributed tax package
WAS on Distributed
Vendor Tax Calculation Package
WAS on Distributed
Vendor Tax Calculation Package
Customer Billing Records
1,000,000+ with fixed completion window

Advanced inbound APIs with asynchronous control

Asynchronous because COBOL is single-threaded and web service call to external tax package is the slowest link. Asynchronous APIs allows COBOL to get program control immediately.

150 connections kept loaded with work and busy

Maximum connections over WOLA to EJB. All 150 loaded up with work requests. COBOL then loops through array to see if response received. If so, then process back results and load that connection with another request. Connections kept fully busy in this manner.

Multi-threaded Java then parallelized web service calls

WAS z/OS and WAS distributed are multi-threaded. Given sufficient processing capacity, the work requests from COBOL may then be handled in a parallel execution fashion.

WebSphere Optimized Local Adapters

Summary

Summary of WOLA Usage and Value

A few points to consider

WOLA is bi-directional, which means you can use it to build solutions that go *outbound* from WAS z/OS, or come *inbound* to WAS z/OS

With CICS and IMS WOLA supports propagation of transaction and security, giving you flexibility to construct solution to your needs

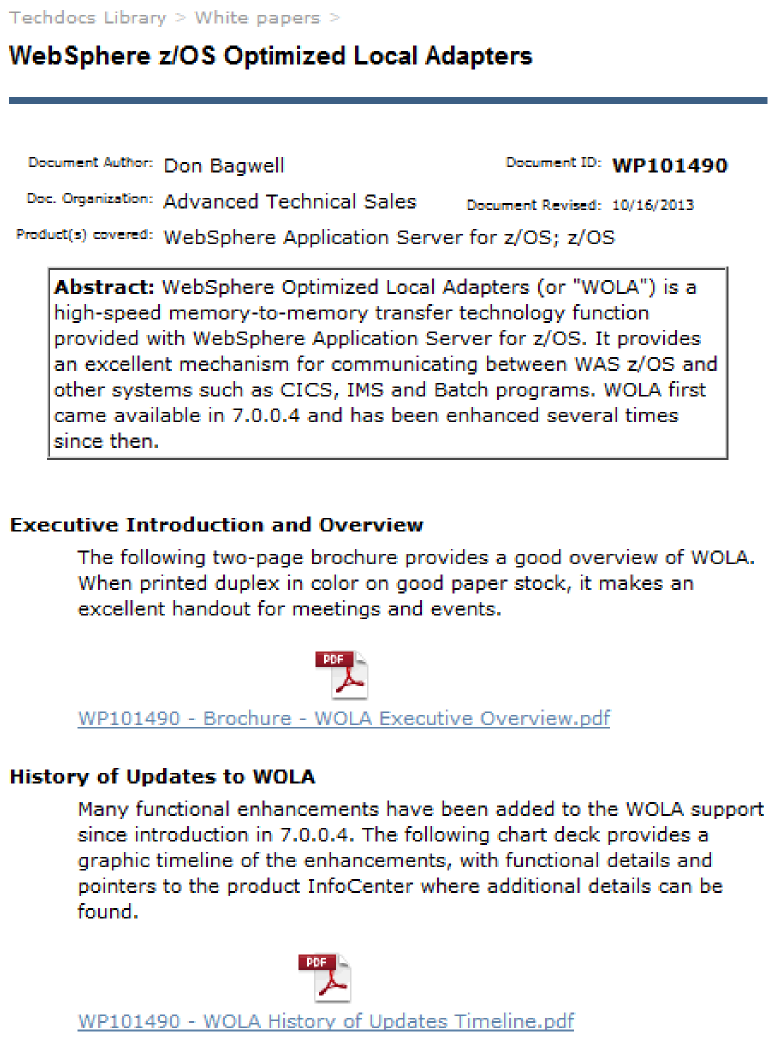
Because WOLA is cross-memory, it provides an extremely low per-call latency profile. In high-volume, repetitive environments latency adds up.

WOLA provides a mechanism to better integrate Java and non-Java, which may allow you to move work to Java and reduce GP usage:

- Move business logic from CICS to WAS z/OS and use WOLA
- Move business logic from batch to WAS z/OS and use WOLA
- Take advantage of ISV solutions in WAS z/OS and integrate with WOLA

WP101490 Techdoc

For more on WOLA, see ibm.com/support/techdocs



Techdocs Library > White papers >

WebSphere z/OS Optimized Local Adapters

Document Author:	Don Bagwell	Document ID:	**WP101490**
Doc. Organization:	Advanced Technical Sales	Document Revised:	10/16/2013
Product(s) covered:	WebSphere Application Server for z/OS; z/OS		

Abstract: WebSphere Optimized Local Adapters (or "WOLA") is a high-speed memory-to-memory transfer technology function provided with WebSphere Application Server for z/OS. It provides an excellent mechanism for communicating between WAS z/OS and other systems such as CICS, IMS and Batch programs. WOLA first came available in 7.0.0.4 and has been enhanced several times since then.

Executive Introduction and Overview

The following two-page brochure provides a good overview of WOLA. When printed duplex in color on good paper stock, it makes an excellent handout for meetings and events.

WP101490 - Brochure - WOLA Executive Overview.pdf

History of Updates to WOLA

Many functional enhancements have been added to the WOLA support since introduction in 7.0.0.4. The following chart deck provides a graphic timeline of the enhancements, with functional details and pointers to the product InfoCenter where additional details can be found.

WP101490 - WOLA History of Updates Timeline.pdf

- Overview material
- History of functional updates
- API coding Primer
- WOLA Videos